WP14 17:00

Proceedings of the 37th IEEE
Conference on Decision & Control
Tampa, Florida USA • December 1998

Control of a Car-Like Robot Using a Virtual Vehicle Approach[1]

M. Egerstedt, X. Hu and A. Stotsky
{magnuse, hu, stotsky}@math.kth.se
Optimization and Systems Theory
Royal Institute of Technology
SE - 100 44 Stockholm, Sweden

Abstract

A solution to the problem of controlling a car-like nonholonomic robot is proposed using a "virtual" vehicle approach, which is shown to be robust with respect to errors and disturbances. The proposed algorithms are model independent, and the stability analysis is done using a dynamical model, in which, for instance, the side slip angles are taken into account.

1 Introduction

In this paper the problem of controlling a car-like robot is studied. Many industrial applications need problems like this to be solved in order to have good and robust path tracking algorithms for different types of mobile robot tasks. Naturally, this has been a well studied topic [2, 5, 10, 11, 9, 3, 4]. A few methods have been proposed to solve the problem, for example, the curvature steering method (see for example [11]) and the flatness approach [4]. However, all these methods use an open-loop control, which is quite sensitive to measurement errors and disturbances, and are model dependent.

In this paper we propose two similar, generic path following control strategies, which are model independent, and use position and orientation error feedback. One strategy gives, in a way that can easily be derived, a direct adjustment of the steady state position error, but has the disadvantage that it only works locally. The other strategy offers a more implicit way for adjusting the steady state position error but works globally.

Our approach can be viewed as a combination of the conventional trajectory tracking, where the reference trajectory is parameterized in time, and the dynamic path following in [9], where the criterion is to stay close to the geometric path, but not necessarily close to an a priori specified point at a given time. In our approach a reference point on the reference path is chosen and a simple control algorithm is used to steer the robot toward that point. What is different from [9] in our approach is that the time evolution of the reference point is governed by a differential equation which contains the position error. One of the advantages of our approach is that it is quite robust with respect to measurement errors and external disturbances. If both errors and disturbances are within certain bounds, the reference point is going to move along the reference trajectory while the robot follows it, otherwise, the reference point might "stop" to wait for the robot. For this reason we call the reference point together with the associated differential equation a virtual vehicle.

In this paper we also analyze the path following control first proposed, by using a dynamic model, instead of a kinematical car model, while we still are working on the stability analysis of the second control algorithm. From Figure 1 one can see that on a plastic floor, even at a fairly low speed (0.2m/s), for a rubber tire mini-car the difference between the dynamic model and kinematic model is significant. Since there are some state variables and coefficients in the model which are difficult to measure in practice, it would not be feasible here to utilize linearization techniques from for example [3, 4], to simplify the analysis.

Once again we emphasize that we design our virtual vehicle in a "closed-loop" fashion, namely, the traveling of the reference point on the reference path does not only dependent on the speed of the robot but also on the robot's current position. Although the focus of the paper is on control of a car-like robot, for the sake of completeness, we also propose a path planning method in which only the nonholonomic kinematic constraints are used. Since our control strategy is model independent, this simplification in path planning seems not very restrictive.

This paper is organized as follows: In section 2 we present our two control algorithms, and in section 3 we present the dynamic model that we, in section 4, use for the stability analysis of the first control algorithm. In section 5, the first controller is implemented on a small car-like robot that shows that our proposed solution does not only work in theory, but also in prac-

[1]This work was partially sponsored by the Swedish Foundation for Strategic Research through its Centre for Autonomous Systems at KTH and partially sponsored by the Swedish Research council for Engineering Sciences

0-7803-4394-8/98 $10.00 © 1998 IEEE

tice, and we also show the results of implementing the second algorithm on a quite different platform. In this case, we choose to work with a Nomad 200 mobile robot in order to stress the fact that our proposed solution is really model independent.

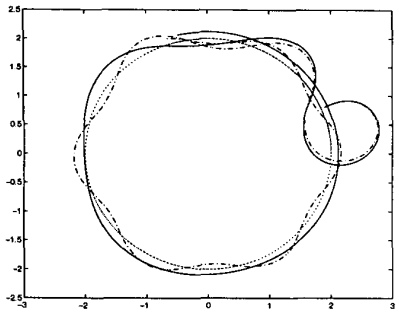


Figure 1: In this figure, the need for a dynamic model when analyzing the performance of a proposed control algorithm is illustrated. A circular parameterized path (dotted) is being tracked, and in the dash-dotted case, the velocities of the robot are derived based on a kinematic model, while the solid path corresponds to velocities derived from a dynamic model.

2 Control algorithms

Our problem is to find a steering angle $\delta_f(t)$ so that the car follows a virtual vehicle $s(t)$ moving on a smooth reference path (i.e. $p'^2 + q'^2 \neq 0 \quad \forall s$)

$$\begin{aligned} x_d &= p(s) \\ y_d &= q(s) \end{aligned} \tag{1}$$

In other words, we require

$$\lim_{t\to\infty} \rho(t) = d \tag{2}$$

$$\overline{\lim_{t\to\infty}} |\psi - \psi_d| \leq \Delta, \tag{3}$$

where

$$\begin{aligned} &\rho(t) = \sqrt{\Delta x^2 + \Delta y^2} \\ &\Delta x = x - x_d, \quad \Delta y = y - y_d. \end{aligned} \tag{4}$$

Here ψ is the yaw angle (orientation of the car), $\psi_d = \arctan \frac{(y-y_d)}{(x-x_d)}$ is the desired orientation, and (x, y) is a reference point on the car, for example the center of gravity or the middle point on the front axle. Furthermore, $\Delta > 0$ is a small number that depends on the maximum curvature of the reference path, and d is the "look-ahead" distance.

2.1 Control Algorithm 1

In order to realize the control aim (2) we define γ and d and require [6]

$$\dot{\rho} - \dot{d} = -\gamma(\rho - d), \tag{5}$$

which implies that

$$\frac{1}{\rho}(\Delta x(\dot{x} - \dot{x}_d) + \Delta y(\dot{y} - \dot{y}_d)) = -\gamma(\rho - d). \tag{6}$$

Taking into account that $\dot{x}_d = \frac{\partial p}{\partial s}\dot{s}$, $\dot{y}_d = \frac{\partial q}{\partial s}\dot{s}$ and solving (6) with respect to $\dot{s}$, we get that

$$\dot{s} = [\Delta x \frac{\partial p}{\partial s} + \Delta y \frac{\partial q}{\partial s}]^{-1} [\Delta x v\dot{x} + \Delta y v\dot{y} + \gamma\rho(\rho - d)]. \tag{7}$$

Assuming that $\Delta x \frac{\partial p}{\partial s} + \Delta y \frac{\partial q}{\partial s} \neq 0$ (it will be zero only if $(\Delta x, \Delta y)^T$ is normal to the curve at (x_d, y_d)), together with (7) gives us s as a function of time, and then $(x_d(s(t)), y_d(s(t)))$ can be calculated. Solving (6) gives us that

$$\rho(t) - d = (\rho(0) - d)e^{-\gamma t} \tag{8}$$

and thus (2) is realized.

Naturally in order for (8) to hold, $\Delta x \frac{\partial p}{\partial s} + \Delta y \frac{\partial q}{\partial s}$ should stay nonzero, and the robot should be steered close to the virtual vehicle. For this we propose the following steering control:

$$\delta_f = -k(\psi - \psi_d), \tag{9}$$

where δ_f is the steering angle, and k should be chosen to reflect the constraint on the maximum steering angle (since $\psi - \psi_d \in [-\pi, \pi]$). Here ψ_d and $(x_d(s(t)), y_d(s(t)))$ are calculated via (7).

2.2 Control Algorithm 2

From (1) we directly get that

$$\begin{aligned} \dot{x}_d &= p'(s)\dot{s} \\ \dot{y}_d &= q'(s)\dot{s}, \end{aligned} \tag{10}$$

which implies that

$$\dot{s} = \frac{p'(s)}{p'^2(s) + q'^2(s)}\dot{x}_d + \frac{q'(s)}{p'^2(s) + q'^2(s)}\dot{y}_d. \tag{11}$$

This suggests that if the car (x, y) tracks the path perfectly we would have

$$\dot{s} = \frac{p'(s)}{p'^2(s) + q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s) + q'^2(s)}\dot{y}. \tag{12}$$

On the other hand, if the velocity of the car $(\dot{x}, \dot{y})^T$ is perpendicular to the tangential direction of the reference path at index s then $\dot{s}$ in (12) would be zero. In order to avoid the situation where the virtual vehicle might get stuck, and make the algorithm work globally, we introduce the following perturbation in $\dot{s}$

$$\dot{s} = kv\rho e^{-\rho/d} + \frac{p'(s)}{p'^2(s) + q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s) + q'^2(s)}\dot{y}, \tag{13}$$

where k and d are two positive constants that need be tuned. The first term in (13) would give the virtual

vehicle a positive "push" when the the real car is approaching it. In this, we also do not allow the virtual vehicle to go backwards, and therefore we set

$$\dot{s} = \begin{cases} \eta & \text{if } \eta \geq 0 \\ 0 & \text{if } \eta < 0, \end{cases} \tag{14}$$

where $\eta = kv\rho e^{-\rho/d} + \frac{p'(s)}{p'^2(s)+q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s)+q'^2(s)}\dot{y}$. For this control algorithm, we also use the same steering control (9) as for the previous algorithm.

Although we are still working on the stability analysis of this algorithm, it is shown, in section 5, that this global approach at least seems to work well in practice as it is implemented on a Nomad 200 mobile platform.

3 Vehicle Model

In order to analyze the control algorithm, we use the so called single track dynamical model [1, 8], which is based both on a description of the balanced forces acting on the vehicle in longitudinal and lateral directions, and on the torque conditions. Although the single track model has its limitations, in a low speed scenario like in our application, it should suffice. If we group the front and the rear wheels together as one single wheel (single track), and let f_x and f_y be the forces acting on the center of gravity of the car, and m_z be the torque, we get a vehicle model that can be seen in Figure 2, where

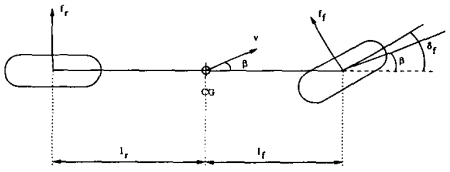


Figure 2: The single track model.

f_f and f_r are the side forces on each wheel, and δ_f is the steering angle of the car. Calculating the forces and the torque gives us that

$$\begin{pmatrix} -mv(\dot{\beta}+r)\sin\beta + m\dot{v}\cos\beta \\ mv(\dot{\beta}+r)\cos\beta + m\dot{v}\sin\beta \\ J\dot{r} \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \\ m_z \end{pmatrix}, \tag{15}$$

where r is the yaw rate, v the longitudinal velocity and β the side slip angle. Furthermore, m is the vehicle mass and J is the moment of inertia.

The tire characteristics of the car can be approximated by

$$\begin{aligned} f_f &= c_f{}^\star\mu(\delta_f - \beta_f) = c_f(\delta_f - \beta_f) \\ f_r &= -c_r{}^\star\mu\beta_r = -c_r\beta_r, \end{aligned} \tag{16}$$

where c_f and c_r are something called "cornering stiffness" of the car, and μ is the so called adhesion coefficient, that depends on what type of surface the road has. In (16), β_f and β_r are front and rear chassis side slip angles respectively.

In order to get an accurate description of what motion it is possible for the car to perform, the following constraints are also needed:

$$\dot{x} = v\cos(\psi+\beta) \tag{17}$$
$$\dot{y} = v\sin(\psi+\beta), \tag{18}$$

where (x, y) is the center of gravity of the vehicle.

3.1 Simplification of Model

Using the assumptions that the velocity of the car is constant, the side slip angle is small, chassis side slip angles are small and that the cornering stiffness is the same for the front and the rear wheels, we get a simplified model of the vehicle that can be written as

$$\dot{x} = v\cos(\psi+\beta) \tag{19}$$
$$\dot{y} = v\sin(\psi+\beta) \tag{20}$$
$$\dot{\beta} + r = a_{11}\beta + a_{12}r + b_{11}\delta_f \tag{21}$$
$$\dot{\psi} = r \tag{22}$$
$$\dot{r} = a_{21}\beta + a_{22}r + b_{21}\delta_f, \tag{23}$$

where

$$\begin{aligned} a_{11} &= (c_r + c_f)/mv \\ a_{12} &= (c_r l_r - c_f l_f)/mv^2 \\ a_{21} &= (c_r l_r - c_f l_f)/J \\ a_{22} &= -(c_r l_r^2 + c_f l_f^2)/Jv \\ b_{11} &= c_f/mv \\ b_{21} &= (c_f l_f)/J. \end{aligned} \tag{24}$$

Here l_f and l_r are the distances between the center of gravity and the front and rear wheels respectively.

4 Stability analysis

In this section we want to show that the first of our proposed control algorithms is in fact a stable one, based on an analysis using the dynamic car model. If we plug in the control algorithm (9)

$$\delta_f = -k(\psi - \psi_d), \tag{25}$$

where $k > 0$, and denote

$$e_\psi = k(\psi - \psi_d) \tag{26}$$
$$e = -\frac{a_{22}}{b_{21}}r - \delta_f \tag{27}$$
$$\delta_f = -e_\psi, \tag{28}$$

then our first step is to present the error model for e and e_ψ. The equation (23) can be written as

$$\dot{e} = a_{22}e - \frac{a_{21}a_{22}}{b_{21}}\beta - \dot{\delta}_f, \tag{29}$$

and equation (22) as

$$\dot{e}_\psi = \frac{b_{21}}{a_{22}}e_\psi - k\frac{b_{21}}{a_{22}}e - k\dot{\psi}_d. \tag{30}$$

Finally, we present equation (21) in terms of e and e_ψ, which gives us

$$\dot{\beta} = a_{11}\beta - \frac{(a_{21}-1)b_{21}}{a_{22}}e + \left\{(a_{21}-1)\frac{b_{21}}{a_{22}} - b_{11}\right\}\frac{e_\psi}{k}.$$

We thus get an error model

$$\dot{X} = AX + b\dot{\psi}_d, \tag{31}$$

where $X^T = (\beta, e_\psi, e)$ and

$$A = \begin{pmatrix} a_{11} & (a_{21}-1)\frac{b_{21}}{a_{22}k} - \frac{b_{11}}{k} & -\frac{(a_{21}-1)b_{21}}{a_{22}^2} \\ 0 & \frac{b_{21}}{a_{22}} & -k\frac{b_{21}}{a_{22}} \\ \frac{a_{21}a_{22}}{b_{21}} & \frac{b_{21}}{a_{22}} & a_{22} - \frac{b_{21}}{a_{22}} \end{pmatrix}$$

$$b = \begin{pmatrix} 0 \\ -k \\ -1 \end{pmatrix}.$$

Straight forward calculations show that $\det(pI - A)$ is a Hurwitz polynomial for all $v > 0$ and $k > 0$, and thus the equation (31) represents a stable dynamics driven by a bounded input if $\dot{\psi}_d$ is bounded. In that case we would have that control aim (3) is realized. Moreover, the positive number Δ can be reduced by reducing the velocity.

Boundedness of $\dot{\psi}_d$

Here, in order to simplify the notation, we only considered the following family of reference paths in our analysis:

$$x_d = s$$
$$y_d = f(s)$$

It is however obvious that the conclusions can be easily extended to the general case.

Evaluating $\dot{\psi}_d$ gives us

$$\begin{aligned} \dot{\psi}_d &= \frac{d}{dt}(\arctan\frac{\Delta y}{\Delta x}) \\ &= \frac{1}{1+\frac{\Delta y^2}{\Delta x^2}} \cdot \frac{(\dot{y}-\dot{y}_d)\Delta x - (\dot{x}-\dot{x}_d)\Delta y}{\Delta x^2}. \end{aligned} \tag{32}$$

Now, taking into account that $\Delta x^2 + \Delta y^2 = d^2$ gives us, after the transients (see (5)),

$$\dot{\psi}_d = \frac{1}{d^2}\{(\dot{y}-\dot{y}_d)\Delta x - (\dot{x}-\dot{x}_d)\Delta y\}. \tag{33}$$

Using that $|\Delta x| \leq d, |\Delta y| \leq d$ together with (7), we can evaluate the bound for (33).

$$|\dot{\psi}_d| \leq \frac{v}{d}[2 + (|\frac{\partial f}{\partial x_d}| + 1)|(\Delta x + \frac{\partial f}{\partial x_d}\Delta y)^{-1}|2d]. \tag{34}$$

From (34) we conclude that $\dot{\psi}_d(t)$ is bounded provided that $\frac{\partial f}{\partial x_d}$ is bounded and that $\Delta x + \frac{\partial f}{\partial x_d}\Delta y$ is bounded away from zero.

We have thus shown that the first of our proposed control algorithms is in fact a stable one.

5 Implementation

5.1 Implementation of the First Control Algorithm

In order to implement the first algorithm, questions concerning robustness, measure and modeling errors, A/D and D/A conversions and numerical complexity need to be addressed before it is possible to get a real system that actually does what it is supposed to. We chose to try our control algorithms on a small, radio controlled car, where we have connected the transmitter to a computer.

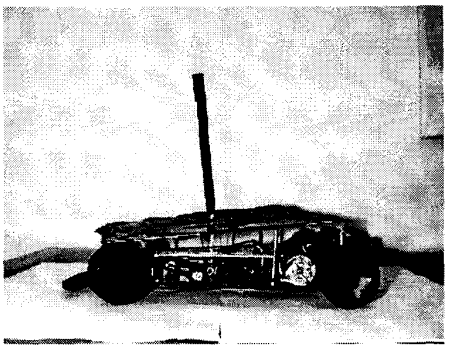


Figure 3: The radio controlled car used for trying out the proposed control algorithms.

However, our car system is based on a fairly cheap toy car with a coarse A/D and D/A conversion as well as a dead zone in the servo system. Therefore the steering is far from precise, so what is working in simulations may not work at all here.

The virtual vehicle is given by $(x_d, f(x_d))$ and are moving along the planned, known trajectory. $\dot{x}_d$ is calculated, as shown in the previous section (7), in such a way that the distance between the actual car and the virtual vehicle converges exponentially to a prespecified distance d. Since we have a sampled system, we have to use some kind of discretized version of the continuous expression, and we just use a simple

first order approximation to calculate the new point $x_d(k+1) = x_d(k) + T\dot{x}_d(k)$. In our system, the frame grabber for the camera, used for tracking the car, sets the sample-interval, T, to be 20ms.

Since the velocity of the car is noise contaminated, we have to to make some kind of estimation, and in our case, it turned out that a straight-forward averaging over a fixed number of sample periods worked sufficiently well.

This gives us all we need in order to determine (x_d, y_d), and the control $\delta_f = -k(\psi - \psi_d)$ that we found in the previous section, can be implemented.

5.2 Path Planning

For the sake of simplicity, we only consider the kinematic constrains of a car-like robot when doing the path planning. Based on [7], we use a type of planner that combines splines with a bang-bang planner. The general idea is that splines are used to plan a path that takes the car close enough to the place where we want to do fine maneuvering, such as parallel parking. We then switch to a different planning mode where we use a bang bang type of planner, using parts of circles, produced by a maximal steering of the car, combined with straight lines. The results from such an approach can be seen in figure 4. One main advantage with this type

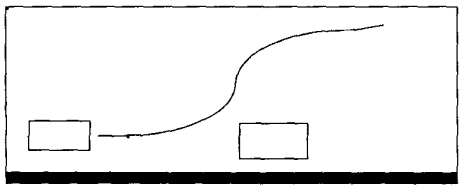


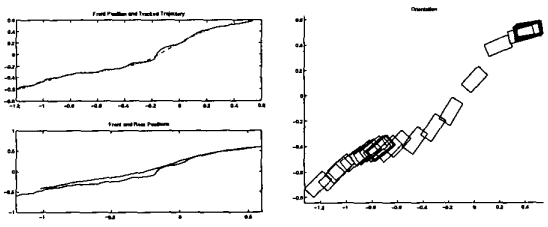
Figure 4: The planned parallel parking path for our actual car-like robot, where the rectangles represent other cars.

of planner is that it is based on algebraic calculations only. We do not need to solve any programming problems and our solutions depend explicitly on the desired safety margins, since the interpolation points can be specified directly, depending on how far away from the obstacles, such as other parked cars, we want to be.

The reason why we chose to use cubic splines as our choice of curves for the free space planner is that they minimize

$$\int_{x_0}^{x_F} f''(x)^2 dx, \tag{35}$$

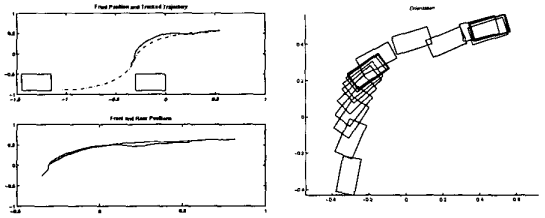
where $f(x)$ is the path that we want the car to follow. This is obviously very useful when the car-like robot has a maximal steering angle constraint.



(a) Car position and the tracked trajectory

(b) Orientation of the car

Figure 5: In the left figure, the tracked trajectory (dotted) and the front point (solid) on the car can be seen, as well as the front and the rear points plotted together. In the left figure, the orientation of the car can be seen.



(a) Car position and the tracked trajectory

(b) Orientation of the car

Figure 6: In the left figure, the tracked trajectory (dotted) and the front point (solid) on the car can be seen, as well as the front and the rear points plotted together. The rectangles corresponds to obstacles, and the picture shows an actual parallel parking experiment. In the left figure, the orientation of the car can be seen.

5.3 Implementation of the Second Algorithm on a Nomad 200

In oder to stress the fact that our proposed control algorithms are really model independent, we choose to implement the second algorithm on a Nomad 200 platform (see Figure 7) instead of on a car-like robot. The dynamics of this platform differ quite a lot from our nonholonomic RC-car, but the approach still seems to work well. This can be seen in Figure 7, where we have plotted different test runs on the Nserver, the Nomad simulator. These test runs, where the robot tracks a circle from different initial positions and configurations, clearly indicates that our proposed second control algorithm works globally in a stable and robust way. It should be emphasized that if we were to use the first, local algorithm on the two first of these test runs, it would fail since the initial positions and orientations in

those cases would make $\Delta x \frac{\partial p}{\partial s} + \Delta y \frac{\partial q}{\partial s} = 0$ and thus $\dot{s}$ would not be defined anymore.

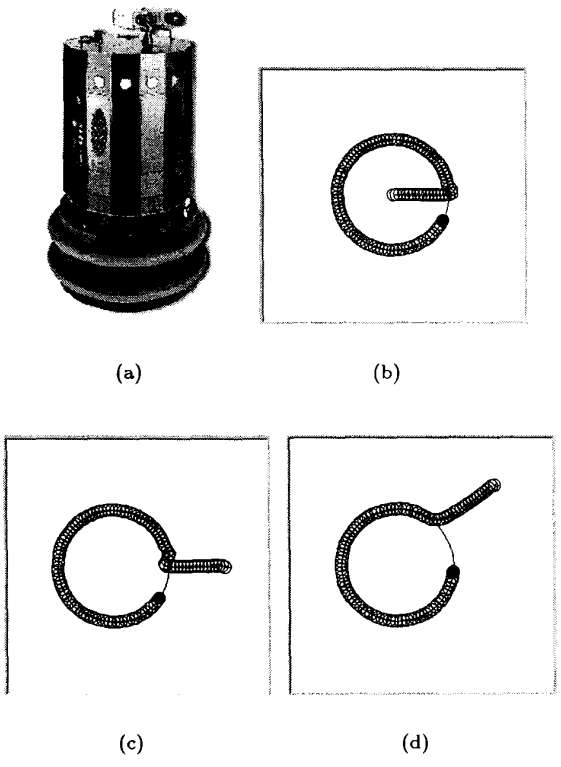


Figure 7: The Nomad 200 and three different initial positions when tracking a circular path.

6 Conclusions

In this paper two model independent path following control strategies are proposed and one of them is analyzed on a dynamical model. What is new here is that by combining the conventional trajectory tracking approach and the more recent geometric path following approach, we design a "virtual vehicle" that moves on the reference path and is regulated in a closed-loop fashion both by the position error and speed.

Implementing these ideas on actual robots gives us some real experimental systems that behave satisfactorily. Some examples can be seen in the Figures 5-7.

References

[1] J. Ackermann: *Robust Control*, Springer-Verlag, London pp 371-375 1993,

[2] R.W. Brockett: Asymptotic stability and feedback stabilization, in *Differential Geometric Control Theory* (Brockett,Millmann and Sussman, eds), pp.181-191, Boston, MA, USA, Birkhauser, 1983.

[3] G. Campion, G. Bastin and B. D'Andréa-Novel: Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 1, Feb. 1996.

[4] M. Fliess, J. Lévine, P. Martin and P. Rouchon: Flatness and Defect of Non-Linear Systems: Introductory Theory and Examples, International Journal of Control, Vol. 61, No. 6, pp. 1327–1361, 1995.

[5] Guldner J. and V. Utkin: Stabilization of nonholonomic mobile robots using Lyapunov functions for navigation and sliding mode control, *Proc. of the 33-rd CDC*, Lake Buena Vista, FL-December 1994, pp.2967-2972.

[6] S.V. Gusev and I.A. Makarov: Stabilization of Program Motion of Transport Robot with Tracklaying Chassis, *Proceedings of LSU*, vol 1, issue 3, No15, 1989.

[7] J-C. Latombe: *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[8] E. Freund and R. Mayr: Nonlinear Path Control in Automated Vehicle Guidance, *IEEE Transactions on Robotics and Automation*, vol 13, No1, Feb 1997.

[9] N. Sarkar, X. Yun and V. Kumar: Dynamic Path Following: A New Control Algorithm for Mobile Robots, *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, Texas, Dec. 1993.

[10] R. Murray and S. Sastry: Nonholonomic motion planning: steering using sinusoids, *IEEE Transactions on Automatic Control*, vol. 38, No 5, pp. 700-716, 1993.

[11] C.C. de Wit: Trends in Mobile Robot and Vehicle Control, *Control Problems in Robotics*, Lecture Notes in Control and Information Sciences 230, pp. 151–176, eds. B. Siciliano and K.P. Valavanis, Springer-Verlag, London, 1998.